基于 GPU 的单幅图像去雾的实现及优化 *

张 津，周祥全，舒 漫，王玉兰†，魏友华，柳炳利

(成都理工大学 数学地质四川省重点实验室，成都 610059)

摘 要：基于暗通道先验规律的去雾算法已取得了良好的去雾效果，但算法所需要的计算时间过长，无法达到实时去雾的要求。使用 GPU 初步并行实现了去雾算法，并确定了算法中需要优化的部分。在优化过程中，一方面将数据存储到高速内存中以实现对数据的快速读取，另一方面设计新的算法实现方式以减少算法的计算量，最终提高了加速比。优化后的加速算法，处理 768×1024 的图像仅需 21 ms，达到了实时去雾的要求。

关键词：图像去雾；GPU；并行优化；实时去雾

中图分类号：TP303　　doi: 10.3969/j.issn.1001-3695.2017.07.0889

Implementation and optimization of single image haze removal based on GPU

Zhang Jin, Zhou Xiangquan, Shu Man, Wang Yulan†, Wei Youhua, Liu Binli

(*Geomathematics Key Laboratory of Sichuan Province*, *Chengdu University of Technology*, *Chengdu* 610059, *China*)

Abstract: The defogging algorithm which based on dark channel prior has achieved good results, but the time spent on computing is too long to meet the requirements of real-time defogging. With parallel GPU, this paper implemented the defogging algorithm and it determined the portion of algorithm which need to optimized. During the optimization process, on the one hand, it stored the data in the high-speed memory to achieve rapid data read; on the other hand, designed a new algorithm implementation to reduce the amount of calculation, it improved the acceleration rate ultimately. The acceleration algorithm just need 21 ms when dealing with images of 768×1024 after optimized, it has reached real-time defogging implementation.

Key Words: image defogging; GPU; parallel optimization; real-time fog removal

0 引言

大气中悬浮雾、霾等类似大气粒子的散射导致场景能见度低，此场景下所拍摄的图像的对比度和颜色的饱和度等都会有一定程度的衰减[1]。而在计算机视觉和图像处理领域，如目标探测、户外监控、遥感图像处理等[2]，都需要能还原真实场景的高质量图像，且在实时导航、自动驾驶等需要实时响应的技术领域中，不仅需要有效地去雾技术，更需要快速地实现技术[3]。

对于去雾技术，追求的是更好的去雾效果、更广的适用范围、更低的算法复杂度[4]，目前比较受推崇的去雾技术是 He 等人[5]提出的基于暗通道先验的去雾方法，该方法有较好的去雾效果以及适用范围，该方法最开始采用的是软抠图（Soft image matting）优化透射率，算法复杂度较高，而后 He 等人[6]采用导向滤波取代了软抠图，而导向滤波的算法复杂度是线性的，并证明了其去雾效果与使用软抠图时基本相同，因此，采用导向滤波的去雾算法成为去雾技术中最有效的算法之一。

去雾技术已日趋成熟，那么现在需要解决的就是如何快速实现。最初使用软抠图时处理一张 600x400 的图像大约需要 10~20 s[6]，使用导向滤波可以缩短计算时间，都仍达不到实时的要求，且对于分辨率更高的图像，其计算速度依然较慢，因此，需要更快地实现技术[7]。自 2007 年 NVIDIA 团队推出了 CUDA 接口，便于研究者使用 GPU 进行研发，基于 GPU 的快速去雾技术也被研究者们实现，如 Lvy 等人[7]使用 GPU 处理 600x400 的图像所需 0.083 s；Xue 等人[8]也使用 GPU 实现了算法，最终处理时间 0.036 s 左右等。计算时间已被缩短，但对于高清图像，加速效果依然不够。因此，本文深入探讨了算法加速的优化部分，从高速内存的高效率使用以及减少算法复杂度两方面入手，进一步提高了算法的加速比，对高清图像基本达到实时处理的效果。

基金项目：国家自然科学基金青年基金资助项目（41602334）；国家自然科学基金面上项目（41672325）；中国地质调查局项目（121201108000150005）；国家重点研发计划课题（2017YFC0601505）；四川省科技厅项目（2017JY0209）

作者简介：张津（1993-），女，山西吕梁人，硕士研究生，主要研究方向为应用数学；周祥全（1992-），男，四川成都人，硕士研究生，主要研究方向为高性能计算；舒漫（1994-），女，四川成都人，硕士研究生，主要研究方向为应用数学；王玉兰（1963-），女（通信作者），四川成都人，副教授，主要研究方向为数学地质、应用数学等（wyl@cdut.edu.cn）；魏友华（1976-），男，四川绵竹人，副教授，硕导，主要研究方向为应用数学；柳炳利（1986-），男，四川成都人，讲师，博士（后），主要研究方向为数学地质.

1　算法简介

目前广泛使用的雾图形成模型是由大气散射模型变形而来[9,10]，即

$$I(x) = J(x)t(x) + A(1 - t(x)) \quad (1)$$

其中：I(x)表示需要去雾的原始图像；J(x)是需要恢复的图像(无雾图像)；A 表示大气环境光；t(x)为透射率。去雾算法就是从I(x)恢复到 J(x)。暗通道先验规律是指：无雾图像的暗通道像素亮度很小，趋近于零[6]。

$$J^{dark}(x) = \min_{c\in\{r,g,b\}}(\min_{y\in\Omega(x)}(J^C(y))) \approx 0 \quad (2)$$

基于暗通道先验规律，才有了这个去雾算法，其算法流程如图 1 所示。

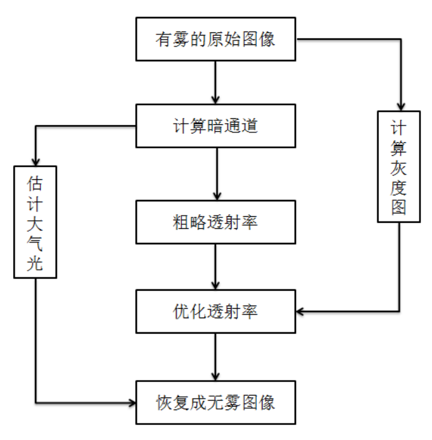


图 1　图像去雾算法流程

具体步骤如下：

a）计算暗通道。设原始图像在 $\Omega(x)$ 的区域内，透射率为常数 $\tilde{t}(x)$，此时的透射率为粗略透射率，则原始图像的暗通道为

$$I^{dark}(x) = J^{dark}(x)\tilde{t}(x) + A(1 - \tilde{t}(x)) \quad (3)$$

b）估计大气光。选取暗通道中亮度最大的 0.1%像素，并根据这些像素点的位置获取原始图像中与之对应的颜色数值，计算器均值作为大气光的估计值 $\bar{A}$。

c）计算粗略透射率。由于 $J^{dark}(x) \approx 0$，可推出：

$$\tilde{t}(x) = 1 - \omega\frac{J^{dark}(x)}{\bar{A}} \quad (4)$$

大气不可能没有任何颗粒[11,12]，彻底清除雾霾，图像可能看起来不自然。因此，引入了一个参数 ω，使图像保留了非常小的雾度，参考文献将其固定为 0.95，表示保留 0.05 的雾度[6,7]。

d）计算灰色图与优化透射率。如果使用粗略透射率还原图像，会出现“块效应”[13]，使用导向滤波算法优化粗略透射率。

$$a_k = \frac{\frac{1}{|\omega|}\sum_{i\in\omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \varepsilon} \quad (5)$$

$$b_k = \bar{p}_k - a_k \mu_k \quad (6)$$

其中：I 为参考图像（He[7]提供的 MATLAB 代码中 I 为原始图像的灰度图，本文也采用同样的方法）；p 为粗略透射率；q 为需要求解的优化后的透射率，在局部领域内 a、b 取平均值，得出

$$\bar{a}_i = \frac{1}{|\omega|}\sum_{k\in\omega_i} a_k ,$$

$$\bar{b}_i = \frac{1}{|\omega|}\sum_{k\in\omega_i} b_k \quad (7)$$

$$q_i = \bar{a}_i I_i + \bar{b}_i \quad (8)$$

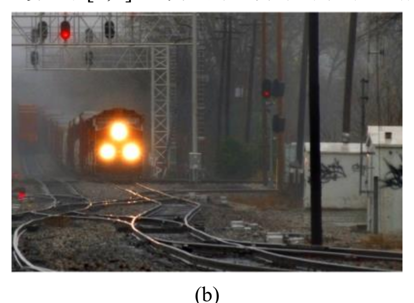
e）复原图像。为避免透射率过小，恢复的时候产生噪声，设定了下限值 t_0 为 0.1，恢复公式为

$$J(x) = \frac{I(x) - A}{\max(t(x), t_0)} + A \quad (9)$$

本文主要介绍算法的加速及优化，具体的算法流程和公式请参考文献[6,7]。本文的去雾效果如图 2 所示。



(a)



(b)



(c)

(d)

图 2　图像去雾效果

图 2(a)(c)为原始图像，即有雾的图像，(b)(d)为去雾后的图像。通过与文献[7]提供的 MATLAB 代码的计算出的像素点数值比较，计算结果基本相同。

2　并行加速和优化

采用 CUDA 平台的 GPU 并行加速，需要了解其硬件体系。CUDA 设备的核心是由一个可扩展的流多处理器（streaming multiprocessors SMs,）整列组成，当调用内核函数时，SM 以线程块（block）为划分单位划分计算资源，同一个 SM 上并发执行多个线程块中的多个线程（thread），每个线程负责计算一个或多个数据，这就是 CUDA 的 SIMT（single instruction，multiple thread）执行模型[14]。

本文加速设计是一个线程执行一个像素点的计算，线程块与线程都以二维的形式，以便于与图像的像素点一一对应，如图 3 所示。

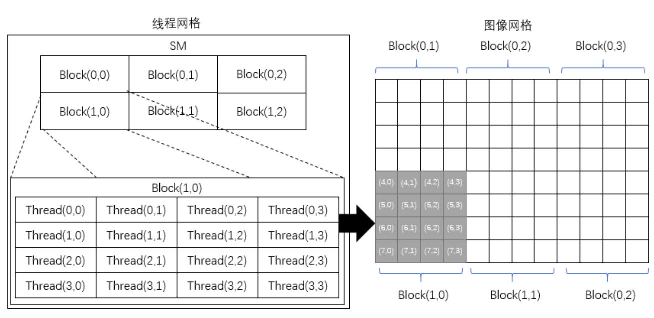


图 3　线程网格与图像网格的对应关系

线程网格初步设计完毕，现在先分析整个算法中哪些步骤需要提高性能，即找出哪些步骤占程序的大部分运行时间，再根据算法复杂度分析其并行性，估计其并行加速效果，如果并行度高，再进一步优化，以提高算法的加速比。

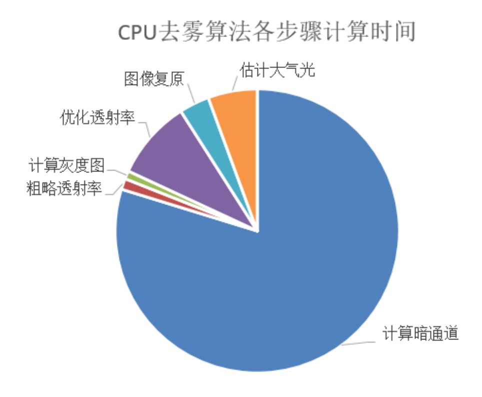


图 4　CPU 去雾算法各步骤计算时间

CPU 去雾算法各步骤计算时间如图 4 所示。由图 4 可看出，整个算法中费时的步骤主要是计算暗通道以及优化透射率，因此，本文主要讨论这两个步骤的加速以及优化，其余步骤基本并行加速即可，对于整个程序的运行时间并没有多大的影响。

2.1　计算暗通道

暗通道的计算量不仅受图像尺寸大小的影响，还与按通道的宽度有关，设暗通道的宽度为 r，图像的尺寸为 m*n，则暗通道的计算量大致为 m*n*r*r*3，除了计算量较大之外，还有一些在实际算法实现中的难点，例如在计算图像边缘的暗通道时，会出现越界现象，如图 5 所示。

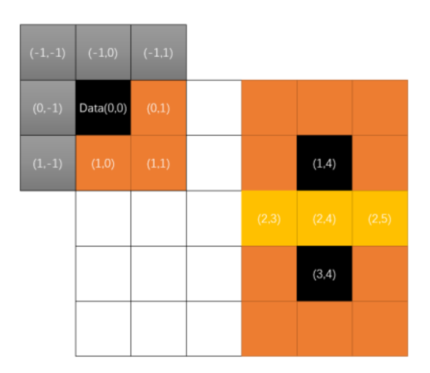


图 5　计算暗通道的难点分析

对于图 5 左上角的第一个像素点（黑色方块），设暗通道的宽度为 3，则有五个像素点属于越界，在实际计算中，需要判断上下左右四个方向是否越界，且计算暗通道是找出通道内的最小值，所以整个计算涉及了大量的判断语句，导致该步骤运行时间过长。

图 5 的右侧，像素点（1，4）和（3，4）在计算暗通道时会出现重复加载数据（图中的三个黄色方块），可以将数据先读取到一个拥有高速带宽的内存中，以减少对全局内存的访问。寄存器是访问最快的内存，但寄存器是每个线程私有的且容量有限[11]，而相邻线程计算所需的数据有部分重叠，因此，使用共享内存，即能满足线程之间数据的重复使用，又能满足对数据的快速读取。

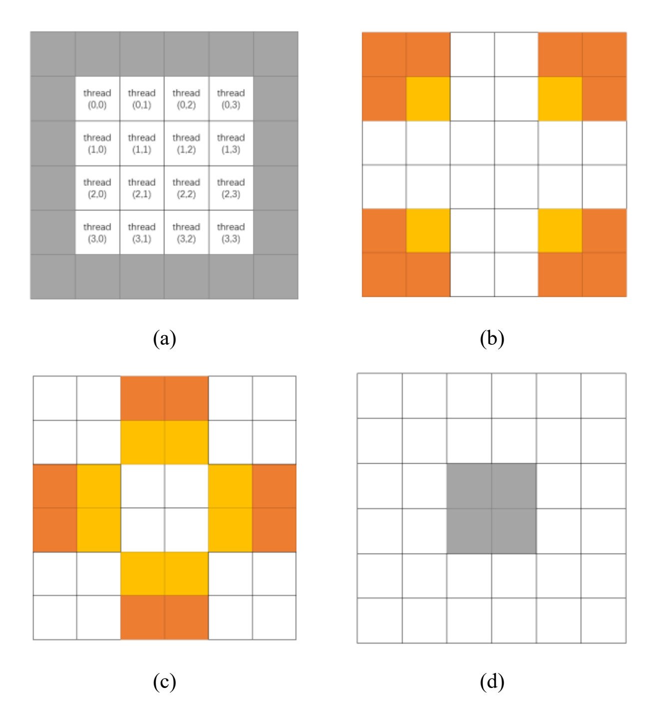


图 6　数据从全局内存到共享内存的读取方式

设暗通道的宽度为 3，图 6（a）中白色区域为一个线程块，该线程块中每个线程与图像的像素点位置一一对应，周围的灰色方块为计算暗通道所需要的“越界”数据，图 6（b）（c）中

黄色方块的线程需要读取包括自身以及周围橙色方块的区域的数据， 6（d）中灰色方块仅需读取自身所在区域的数据，通过这种方式读取，可以将该线程块中所有线程计算所需的数据存储在共享内存中。但对于全部数据，不同的线程块也会出现重复读取。

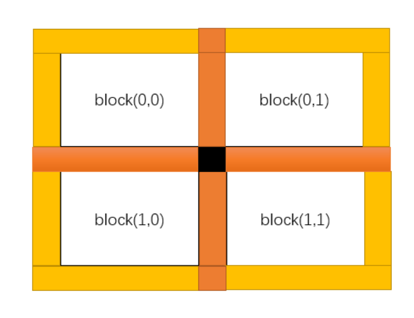


图 7　不同线程块之间的数据的重复读取

每个线程块读取的数据如图 6（a）所示。不同线程块之间的数据的重复读取如图 7 所示。再根据图 7 可知，相邻线程块对于“越界”数据是会出现重复读取的，黄色区域为读取 1 次，橙色区域表示读取 2 次，而黑色区域为 4 次。但总体而言，使用共享内存减少了对全局内存的访问。

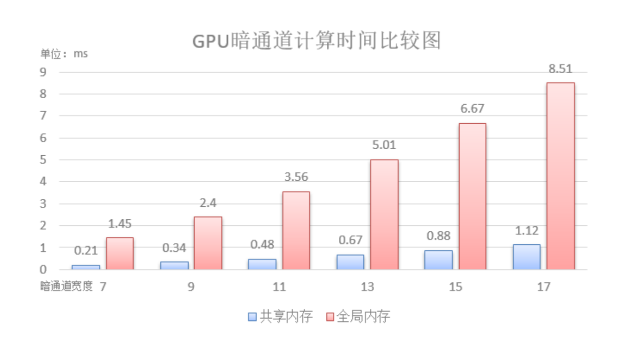


图 8　使用共享内存与全局内存的 GPU 暗通道计算时间对比

使用共享内存与全局内存的 GPU 暗通道计算时间对比如图 8 所示。根据图 8 可看出，使用共享内存，能有效地减少算法的计算时间，使用共享内存后计算速度平均为使用全局内存的 7 倍，充分利用高速内存，有效地提高加速比。

2.2　优化透射率

优化透射率的公式为式（5）～（8）。若直接根据公式加速算法，通过合并核函数后仅需启动两个核函数即可，但需要考虑边界、共享内存的大小、合并访存等问题，而导向滤波则需要启动二十多次内核函数，且多次调用 box_filter 函数。为此，作了多次实验，分析了在不同的窗口大小的情况下，两种算法计算所需的时长。

两种优化透射率方法计算时长的比较如图 9 所示。由图 9 可看出，随窗口宽度变大，使用 box_filter 函数的优势越大，所以本次实验使用 box_filter 函数。box_filter 函数在并行中分为两个部分，一部分是行中实现累加，另一部分是在列中实现累加，其余的都是一些简单的矩阵运算。因此，优化并行加速就得提高累加效率。

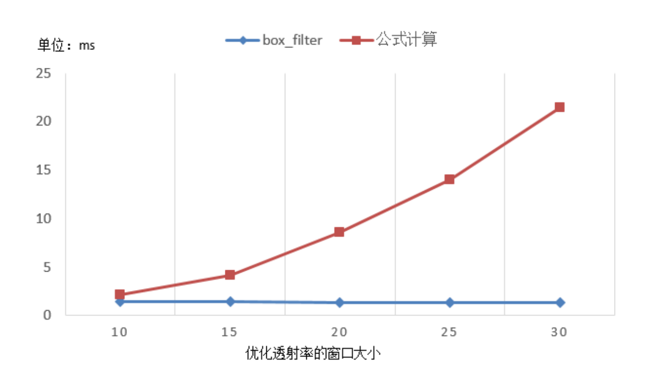


图 9　两种优化透射率方法计算时长的比较

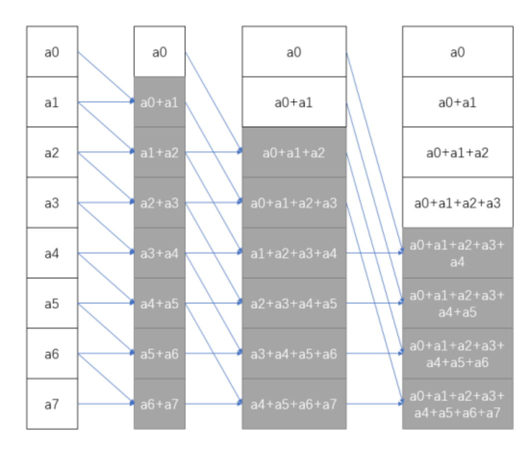


图 10　累加的一般并行方法

图 10 显示了累加的初步加速方法，只有 8 个数据时也需要进行 17 次加法运算。当数据量为 N 时，需要进行(N-1)+(N-2)+(N-4)+...+N/2 次，即 $N\log_2(N)$-(N-1)次运算，计算效率太低了，随着 N 的增大，所需要的计算时间也急剧增长。

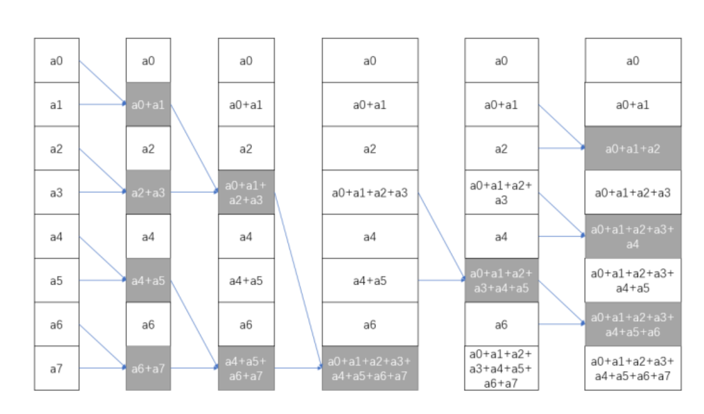


图 11　优化后的并行累加

图 11 为优化后的并行累加。可以看出，同样是 8 个数据，一共只需要 11 次运算，依然假设数据量为 N，则优化后的累加算法需要(N-1)+(N/2-1)+…+(4-1)+(2-1)，即 2N-$\log_2(N)$-2 次运算，算法复杂度降低了很多。

初始版本与优化版本的计算量比较如图 12 所示。优化版本的计算量是呈线性增长的，根据图 12 可看出，相比于初始版本，计算 1 024 个数据时仅需要四分之一不到的计算资源。因此，随着图像尺寸的增大，加速效果也会越来越明显。

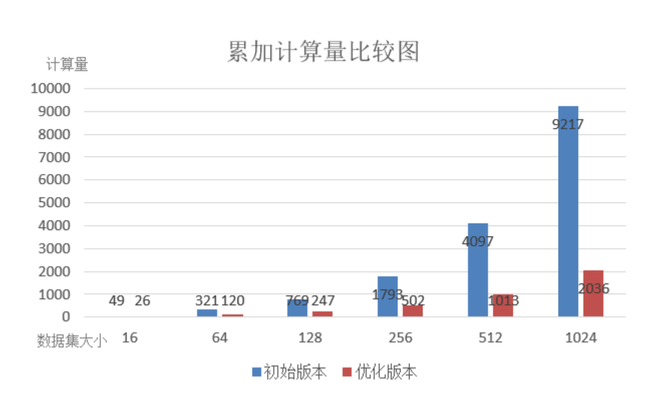


图 12　初始版本与优化版本的计算量比较

3　实验结果

与本次实验相关的环境如表 1 所示，程序运行版本为 release。

表 1　实验环境

硬件环境：	
主机端：	AMD FX(tm)-6300 Six-Core Processor 3.50GHz 8.00GB 内存
设备端：	GeForce GTX970 GPU，4GB 显存
软件环境：	
主机端：	Microsoft Windows 10(64bit Edition)， Microsoft Visual Studio 2013
设备端：	CUDA Toolkit 7.5

在此环境下，实验了不同尺寸的图像，其效果如表 2 所示。

表 2　不同尺寸图像的计算时间

图像尺寸	CPU 版本 /ms	GPU 初始版本 /ms	GPU 优化版本 /ms
300*400	220.41	32.13	4.3
400*600	431.57	60.06	7.36
600*800	823.18	127.53	14.18
768*1024	1269.24	173.72	20.91

由表 2 可看出，优化后的版本工作效率大约是优化前的十倍左右，对于 768*1024 高分辨率图像，也仅需 21 ms 不到，也就是说近 50 帧每秒，这已经达到实时处理的要求了。

4　结束语

国内外已有不少对去雾算法的加速研究，本文对于 600*400 的图像的处理速度确实优于前人，然而因为不同的实验环境，实在难以比较哪种方案的优劣。而本文的重点是提出了对于加速优化的一些建议，主要是以下两点：

a)充分利用硬件设施中的高速内存，减少对全局内存的访问。如果该数据是同一个线程块中的不同线程需要重复使用的，可以存储到共享内存中，如果数据量小，且是线程私有的，可放入寄存器。

b)降低算法的复杂度，有效地减少计算量。同一个算法也会有不同的实现方式，尽量在保证算法并行度的同时降低复杂度。

对于去雾算法的并行优化，下一步工作是：数据存储到共享内存中时，如何以合并访存的方式对全局内存进行访问；在优化透射率的时候减少内核函数的启动开销。

参考文献：

[1] Narasimhan S G, Nayar S K. Contrast restoration of weather degraded images [J]. IEEE Trans on Pattern Analysis Machine Intelligence, 2003, 25 (6): 713-724.

[2] Liu Qi, Gao Xinbo, He Lihuo, et al. Haze removal for a single visible remote sensing image [J]. Signal Processing, 2017, 137: 33-43.

[3] Hung C L, Ma Zhaohui, Lin Chunyuan, et al. Image haze removal of optimized contrast enhancement based on GPU [J]. Frontier Computing, 2016, 375: 53-63

[4] 郭璠，蔡自兴，谢斌，等. 图像去雾技术研究综述与展望 [J]. 计算机应用, 2010, 30 (9): 2417-2421.

[5] He Kaiming, Sun Jian, Tang X. Single image haze removal using dark channel prior [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2011, 33 (12): 2341-2353.

[6] He Kaiming, Sun Jian, Tang Xiaoou. Guided image filtering [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2013, 35 (6): 1397-1409.

[7] Lvy Xingyong, Chen Wenbin, Shen I F. Real-time dehazing for image and video [C]// Proc of the 18th IEEE Conference on Pacific Computer Graphics and Applications. 2010.

[8] Xue Y G, Ren Ju, Su Huayou, et al. Parallel implementation and optimization of haze removal using dark channel prior based on CUDA [J]. High Performance Computing, 2013, 207: 99-109.

[9] 李佳童，章毓晋. 图像去雾算法的改进和主客观性能评价 [J]. 光学精密工程, 2017, 25 (3): 735-741.

[10] 魏颖慧，张彦娥，梅树立，等. 基于暗通道先验和区间插值小波变换的图像去雾霾方法 [J]. 农业工程学报, 2017, 33 (S1): 281-287.

[11] 宋颖超，罗海波，惠斌，等. 尺度自适应暗通道先验去雾方法 [J]. 红外与激光工程, 2016, 45 (9): 286-297.

[12] 陈书贞，任占广，练秋生. 基于改进暗通道和导向滤波的单幅图像去雾算法 [J]. 自动化学报, 2016, 42 (3): 455-465.

[13] Tarel J P, Hautiere N. Fast visibility restoration from a single color or gray level image [C]// Proc of the 12th IEEE International Conference on Computer Vision. 2009.

[14] Nicholas Wilt. 2014. CUDA 专家手册——GPU 权威编程指南 [M]. 北京：机械工业出版社.